

The Multimission VICAR Planner: A Fielded Planning System for Scientific Data Analysis

Steve A. Chien and Helen B. Mortensen
Jet Propulsion Laboratory
California Institute of Technology
4800 oak Grove Drive hi/S 525-3660
Pasadena, CA 91109-8019
{steve.chien,helen.mortensen}@jpl.nasa.gov

Abstract. The Multimission VICAR Planner (MVP) system is an AI planning system which constructs executable image processing programs to support Operational Science Analysis (OSA) requests made to the Jet Propulsion Laboratory (JPL) Multimission Image Processing Subsystem (MIPS). MVP accepts as input: image files and a high-level specification of desired corrections, enhancements, output properties (such as for mosaics). MVP then derives: unspecified but required processing steps, relevant image processing library programs, and appropriate parameter settings for such programs - constructing an executable image processing program to fill the image processing request. MVP is currently available to analysts to fill requests and reduces the effort to fill radiometric correction, color triplet reconstruction, and mosaicking tasks by over an order of magnitude.

1 introduction

In recent times, improvements in spacecraft imaging hardware have caused a massive increase in the amount of scientific data and variety of science data products. Simultaneously, increased sophistication of image processing algorithms has complicated the image processing task. While ensuring physical access to the vast amounts of space-related data can be achieved, it is often extremely difficult for the average user to manageably prepare and process the available scientific data.

One method for reducing this data access, preparation, and analysis problem is the development of general purpose data processing languages and interfaces. These languages and interfaces allow users to access and process data within a common environment. For image processing, the VICAR environment (Video Image Communication and Retrieval)¹ [Lavoie et al. 1989] is a major constituent of JPL's image processing

⁰This work was performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. Other past and present members of the MVP team are Christine Ying, Shouyi Hsiao, Darren Mutz, Alex Gray, Joc Nieten, and Jean Lorré.

¹This name is somewhat misleading as VICAR is used to process considerable non-video image data such as MAGELLAN synthetic aperture radar data.

capability. VICAR provides a standard interface to allow a user to retrieve video image data and apply sophisticated image processing algorithms. The principal focus of the VICAR system is planetary imaging, and as such it supports imaging for JPL flight projects including VOYAGER, VIKING, MAGELLAN, GALILEO, CASSINI, etc. VICAR has been applied to other space imaging missions such as IBIS and LANDSAT. The VICAR system has also been applied to numerous other applications including: astronomy, earth resources, land use, biomedicine, and forensics. VICAR is a principal component of the Multimission Image Processing Laboratory (MIL). Outside of JPL, VICAR users include universities, the military, research institutions, aerospace corporations, companies, and Galileo III IPS (home institution image processing subsystem) sites with a total user group of over 100 users.

VICAR allows individual image processing steps (called VICAR programs) to be combined into more complex image processing scripts called procedure definition files (PDL's). As one of their primary duties, JPL analysts construct PDL's to perform image correction, image enhancement, construct mosaics, and to create movies and render objects. Individual processing programs perform functions such as:

1. photometric correction - correcting the image for lighting conditions due to the position of the sun relative to the imaging device and target,
2. radiometric correction - correcting for varying camera response depending on camera state and other properties such as where in the field of view the image is read,
3. line fillin - interpolating missing lines caused by data transmission errors.

For example, shown in Figure 1 is a code fragment to perform portions of image navigation manually using a Galileo image². The higher-level conceptual steps are shown at the left, and the corresponding VICAR code at the right. In this case, the tasks being accomplished are acquiring initial navigation information, constructing initial overlap pairs, refining initial overlap pairs, checking for a previous tiepoint file, manually generating or refining tiepoints, and constructing the OM matrix for image navigation. In this case the overall user goal is to navigate the image. The other subgoals (and steps) are necessary to support this goal due to the dependencies of VICAR and image navigation.

In order to fulfill OSA requests for image processing, analysts must create complex VICAR programs, determining relevant programs, order of execution, and parameter settings using their knowledge of the processing steps and processing program requirements.

Unfortunately, manual construction of VICAR programs is both labor and knowledge intensive. The VICAR procedure generation problem is also a knowledge intensive task in that an analyst must possess knowledge of:

1. image processing and image processing programs (as of 1/93 there were approximately 50 frequently used programs, some having 10s options)
2. database organization and database label information to understand the state of relevant data
3. the VICAR programming language to produce and store relevant information.

²This code was generated by MVP.

| Conceptual Steps | VICAR Code |
|---|---|
| get initial navigation information | IBISNAV OUT="file_list\NAV" PLANE1=target_0_10 + PROJECT="GILL" SEDR=@RIMSAC FILENAME="file_list\list" |
| construct initial overlap pairs | ! Construct initial overlap pairs MOSPLOT MOSPLOT In="file_list\NAV" nlines 0_6 ncs=samples_0_6 project="GILL" ! mos overlap is just a holder for the overlap plot dcl copy print on xplotmos - verlap dcl print not eod mos overlap |
| refine initial overlap pairs | !! Refine initial overlap pairs edibis EDIBIS INP="file_list\OVER" |
| find previous tiepoint file (if present) | !! Manmatch mosaic file list !! If there is no existing tiepoint file !! Check if a tiepoint file exists !! The following codes in written VMS !! LOCAL SIR STRING INIT = " LE1 _ONFAIL = "CONTINUE" !! Allow the pdf to continue !! If a file is not found DCL DEASSIGN NAME DCL DEFINE NAME "F\$SEARCH("file_list\TP") LOCAL STRING TRANSLOG NAME STR LE1 _ONFAIL = "RETURN" !! Set PDF to return on error |
| use manmatch program to construct or refine tiepoint file | IF (STR = "") MANMATCH INP="file_list\NAV" file_list\OVER" + OUT="file_list\TP" PROJECT="GILL" SEDR FILENAME="file_list\LIST" |
| use tiepoints to construct OM matrix | !! If an old tiepoint file exists !! If 1 tiepoint file is part of input and 1, 1st overwritten ELSE MANMATCH INP="file_list\NAV" file_list\OVER" file_list\TP" + OUT="file_list\TP" PROJECT="GILL" SEDR FILENAME="file_list\LIST". !OMCOR2 OMCOR2 INP="file_list\NAV" file_list\TP" PROJECT="GILL" GROUND=@GOOD OMCOR2 INP="file_list\NAV" file_list\TP" PROJECT="GILL" GROUND=@GOOD |

Figure 1: Sample VICAR Code Fragment

1 Because of the complexity and amount of program knowledge relevant to the task as well as the many interacting problem goals, VICAR procedure generation is a labor intensive task. Generation of a complex VICAR procedure may take up to months of analyst time.

One difficulty facing analysts is the diversity of knowledge required to produce expert VICAR procedures. While certain VICAR users, such as expert analysts, may possess much of this knowledge, the vast majority of VICAR users are novice to one or more aspects of this knowledge. Unfortunately, this increases the difficulty of data access and preparation and increases the load on experts who must spend a significant amount of their time assisting those less knowledgeable. For example, a university user may know a great deal about the science behind the imaging and the theory behind the processing steps but may know little about the underlying assumptions of the implementation of the processing steps or VICAR itself. Similarly, a programmer who writes processing programs may know quite a bit about their particular program but may experience difficulty in writing a VICAR procedure to generate data to test his or her program. This great need for VICAR knowledge exists because of the significant time it takes to become proficient in multiple aspects of VICAR. Generally, a VICAR user with 1-2 years of experience is considered a novice VICAR user, while it may take 4-5 years to become a VICAR expert.

2 Ovcl'view

MVP [Chien 1994a, Chien 1994c] partially automates generation of image processing procedures from user requests and a knowledge-based model of an image processing area using Artificial Intelligence (AI) automated planning techniques [Iwasaki and Friedland 1985, Pemberthy and Weld 1992, Stefik 1981]. In AI planning, a system uses: 1) a model of actions in a domain; 2) a model of the current state; and 3) a specification of the desired state: to reason about what actions to take to achieve some specified

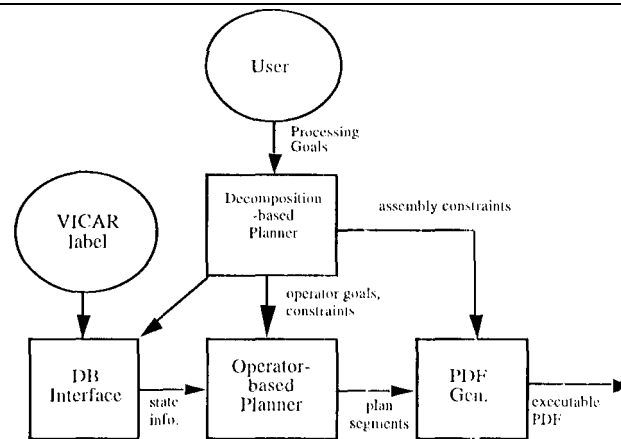


Figure 2: MVP System Architecture

goals. In VICAR image processing the actions are VICAR image processing programs, the current state is the current state of the image files of interest, and the specification of the desired State corresponds to the user image processing goals. By partially automating the filling of basic science image processing requests, image processing request turnaround time will be reduced, analysts time will be freed for more complex and challenging science requests, and analyst workload will be reduced.

2.1 The MVP Architecture

The overall architecture for the MVP system is shown in Figure 2. The user inputs a problem specification consisting of processing goals and certain image information using a menu-based graphical user interface. These goals and problem context are then passed to the decomposition-based planner. The decomposition-based planner uses image processing knowledge to classify the overall problem type which the user has specified in a process called *skeletal planning* [Wasaki and Friedland 1985]. This classification is then used to decompose the problem into smaller subproblems in a process called *hierarchical planning* [Stefik 1981]. The subproblems produced by the decomposition process are then solved in a process called *operator-based planning* [Pembertly and Weld 1992], in which a planner uses a description of possible actions (in this case image processing steps) to determine how to achieve subproblem goals as indicated by the problem decomposition. The resulting plan segments are then assembled using constraints derived in the decomposition process. The resulting plan is then used to generate an actual executable VICAR PDF using conventional code-generation techniques.

MVP uses both decomposition and operator-based planning paradigms for two reasons: search control and user understandability. Plans in the MVP domain can be of considerable length (up to 100 steps) and each step (or VICAR program) can involve reasoning about numerous complex effects (many operators have tens of effects). Due to the large search space caused by this complexity, conventional operator-based planning approaches are not able to tractably construct plans in the VICAR domain without significant control knowledge. By using the decomposition planning paradigm, MVP breaks up the large search space planning problems caused by the complexity of the image processing problems into multiple smaller problems thus reducing the search

problems encountered during operator-based planning. Indeed, the problem decomposition rules used in MVP can be considered a very important form of search control knowledge essential to MVP's image processing capability.

MVP also uses decomposition-based planning for reasons of user understandability. Even if a purely operator-based planning approach were able to generate plans to solve the VICAR problems, these plans would be difficult for MIL analysts to understand because MIL analysts do not consider an entire image processing problem all at once. Typically, analysts begin by classifying the general problem being addressed into one of a general class of problems, such as mosaicking, color triple processing, etc. They then use this classification and the problem context to decompose the plan into several abstract steps, such as local correction, navigation, registration, touch-ups, etc. Because MVP uses decomposition-based planning to reduce the original image processing problem, it can easily produce an annotated trace of how the overall problem was classified and decomposed, simplifying analyst understanding of the plan generation process.

2.2 Skeletal and Hierarchical Planning Using Decompositions

MVP integrates decomposition and operator based planning paradigms. MVP first, reduces a problem using decomposition methods, then solves the resulting subproblems using operator planning techniques. MVP uses knowledge represented as decomposition rules to perform skeletal and hierarchical planning.

2.2.1 Skeletal and Hierarchical Planning in MVP

Skeletal planning [Iwasaki and Friedland 1985] is an approach to planning which casts planning as a structured classification problem. In skeletal planning, a planner identifies a new problem as one of a general class of problems based upon the goals and initial state. This technique was originally developed as a model of experiment design in molecular biology; however, skeletal planning is also an accurate model of how expert analysts attack VICAR procedure generation problems. Typically, in a VICAR problem, there is a central goal for processing, such as mosaicking, which then dictates a decomposition of the overall problem into subproblems such as local correction, navigation, and registration. MVP attacks a VICAR problem by first determining the general problem class, and then using this problem class to perform an initial decomposition of the top-level image processing goals.

Hierarchical planning [Stefik 1981] is an approach to planning where abstract goals or procedures are incrementally refined into more and more specific goals or procedures as dictated by goal or procedure decompositions. MVP uses this approach of hierarchical decomposition to refine the initial skeletal plan into a more specific plan specialized based on the specific current goals and situation. This allows the overall problem decomposition to be influenced by factors such as the presence or absence of certain image calibration files or the type of instrument and spacecraft used to record the image. For example, geometric correction uses a model of the target object to correct for variable distance from the instrument to the target. For VOYAGER images, geometric correction is performed as part of the local correction process, as geometric distortion is significant enough to require immediate correction before other image processing steps can be performed. For WCVY, for GALILEO images, geometric correction is postponed until the registration step, where it can be performed more efficiently.

| LHS | RHS |
|----------------------------------|----------------------------------|
| G_i = initial goal set/actions | G_R = reduced goal set/actions |
| C_0 = constraints | C_1 = constraints |
| C_2 = context | N = notes on decomposition |

Figure 3: Decomposition Rule Syntax

| LHS | RHS |
|---|---|
| G = mosaicking goal present | G_R = 1. local correction, 2. navigation 3. registration 4. mosaicking 5. touch-ups |
| C_0 = null | C_1 = these subtasks be performed in order 1,2,3,4,5. |
| C_2 = an initial classification has not yet been made | protect local correction until mosaicking |
| | N = problem class is mosaicking |

Figure 4: Skeletal Planning Decomposition Rule

2.2.2 Decomposition-based Planning in MVP

MVP uses a decomposition approach [Lansky 1993, Erolet al. 1994] to perform Skeletal and 1 hierarchical planning. In a decomposition approach, decomposition rules dictate how to break a problem into smaller problems. In many cases, it is possible to decompose a problem in several ways. In these cases, the planner then searches the space of possible decompositions. Decomposition approaches are extremely powerful in that many other paradigms (such as modal truth criterion planning [Lansky 1993]) can be implemented in a decomposition-based approach. The syntax for a decomposition rule is shown in Figure 3.

This rule states that a set of goals or actions G_i can be reduced to a new set of goals or actions G_R if the set of constraints C_0 is satisfied in the current plan and the context C_2 is satisfied in the current plan provided the additional constraints C_1 are added to the plan. Skeletal planning in MVP is implemented by encoding decomposition rules which allow for classification and initial decomposition of a set of goals corresponding to a VICAR problem class. The LHS of a skeletal decomposition rule in MVP corresponds to a set of conditions specifying a problem class, and the RHS specifies an initial problem decomposition for that problem class. For example, Figure 4 shows a decomposition rule for the problem class *mosaicking with absolute navigation*.

The simplified decomposition rule shown in Figure 4 states that if mosaicking is a goal of the problem and an initial problem decomposition has not yet been made, then the initial problem decomposition should be into the subproblems local correction, navigation, etc. and that these steps must be performed in a certain order. This decomposition also specifies that the local correction goals must be protected during

| LHS | RHS |
|--|--------------------------------|
| $G =$ navigation action present | $G_s = 1.$ absolute navigation |
| $C_0 =$ null | $C_1 =$ null |
| $C_2 =$ the project is VOYAGER or GALILEO and limb present in all images | $N =$ null |

Figure 5: Hierarchical Refinement Decomposition Rule

the navigation and registration processes. In general, MVP permits goals and abstract steps to be specified in the G_L and G_R fields. The constraints C_0 and C_1 and context specify restrictions on when the rule is applicable, and include: 1. constraints on the ordering of steps or goals; 2. constraints on the assignment of variables representing objects in the plan; 3. goals or steps that must be present in the plan; and 4. goals or steps that are not allowed to be present in the plan. Hierarchical planning is also implemented within the decomposition framework. In this case the LHS specifies a context, in which a set of goals or actions can be decomposed into a lower level set of goals or actions. For example, the decomposition rule in Figure 5 states that if the limb is present in all of the images (meaning that the sun-facing edge of the planet is visible in all of the images), for VOYAGER and GALILEO images, the navigation step can be performed by absolute navigation (a process in which each of the images can be navigated independently).

This decomposition-based approach to skeletal and hierarchical planning in MVP has several strengths. First, the decomposition rules very naturally represent the manner in which the analysts attack the procedure generation problem. Thus, it was a relatively straightforward process to get the analysts to articulate and accept classification and decomposition rules for the subareas which we have implemented thus far. Second, the notes from the decomposition rules used to decompose the problem can be used to annotate the resulting PDF to make the VICAR programs more understandable to the analysts. Third, relatively few problem decomposition rules are easily able to cover a wide range of problems and decompose them into much smaller subproblems.

2.3 Operator-based Planning in MVP

MVP uses classical operator-based planning techniques to solve subproblems produced by the decomposition-based planner. An operator-based planner uses:

1. a model of actions M (in this case the model represents the requirements and effects of individual VICAR steps);
2. a specification of a current state C (this corresponds to the current database state); and
3. a specification of a goal criteria G (this corresponds to user request specification)

to derive:

a sequence of actions A , that when executed in the current state C , result, in a state which satisfies the goal criteria G . In this case A will correspond to the VICAR script, the user can run the script to see the results of the actions. The user can also see the results of the actions in the VICAR script.

In operator-based planning, an action is represented in terms of its preconditions (those things required to be true before an action can be executed), and its effects (those things true after an action is executed). For example, in VICAR image processing, the program GALSOS is used to radiometrically correct Galileo image files. This would be represented by a planning action for the GALSOS program, which could be applied to an image file. This action would have the precondition that the image file be a Galileo image file. This action would also have the effect that the image file is radiometrically corrected after GALSOS has been run.

When constructing a plan to achieve a goal G1, a planner will consider those actions which have G1 as an effect. Thus, if the planner wanted to achieve that a particular image file was radiometrically corrected, it would consider applying the VICAR program GALSOS on the image file. If a planner decides to add an action A1 to a plan to achieve a goal, it will then have to achieve all of the preconditions of A1. This process is called *subgoaling*. For example, the VICAR program PTP requires that an image file be in byte format before PTP can be applied. Thus if the planner decides that it wants to apply the PTP program to a file, it then must ensure that the image file is in byte format. In some cases this will already be true, in other cases running a program to change the file format may be required.

Planning is also complicated by the fact that there are typically interactions between subparts of the plan. Thus, actions introduced to achieve goals in one part of the plan may undo goals achieved in another part of the plan. The process of ensuring that such interactions do not occur is called *protection*. Protection can involve such measures as ensuring that the goal is no longer needed when it is undone, or ensuring that the offending action effect does not in fact refer to the same object as the achieved goal (by creating a copy of a file, for example). We have only briefly sketched some of the elements of operator-based planning, for a more detailed treatment of operator-based planning algorithms the reader is referred to [Pembertly and Weld 1992, Chapman 1987].

To illustrate the operator-based planning process, consider the (simplified) image processing operators shown in Figure 6. This information can be summarized by the information shown below indicating the relevant programs for achieving the goals of missing line fillin, spike removal, and radiometric correction for Voyager and Galileo images. When constructing a plan to achieve these goals, depending on the project of the image file (e.g., either Voyager or Galileo), MVP can determine the correct program to use because the preconditions enforce the correct program selection.

| | Voyager | Galileo |
|----------------------|-----------|-----------|
| fillin missing lines | VGRFILLIN | GILFILLIN |
| remove spikes | ADESPIKE | ADESPIKE |
| radiometric corr. | FICOR77 | GALSOS |

However, determining the correct ordering of actions can sometimes be complex. In this case, the correct order to achieve the goals of line fill in, spike removal, and radiometric correction is dependent upon the project of the file. In the case of Voyager files, ADESPIKE (spike removal) requires raw pixel values and FICOR77 (radiometric) changes pixel values to correct for camera response function, so FICOR77 removes a necessary condition for ADESPIKE. This interaction can be avoided by enforcing that, ADESPIKE occurs before FICOR77. VGRFILLIN requires binary HDR header on the image file which is not maintained by ADESPIKE. This interaction can be avoided by

| Operator | VGRFILLIN | GLLFILLIN | ADESPIKE | FICOR77 | GALSOS |
|---------------|-----------------------------------|-----------|---|--|---|
| Preconditions | VGR image EDR present | GLL image | (GLL image) or ((VGR image) and (raw values)) | VGR image | GLL image raw pixel values |
| Effects | missing lines filled in | | spike removal not raw values | radiometric corr. blemish removal not raw values | radiometric corr. reed-solomon overflow corr. saturated pixel corr. not missing line fillin |

Figure 6: Simplified Operator Definitions

requiring VGRFILLIN to be executed before ADESPIKE.

The Galileo case is slightly different. GALSOS undoes missing line fillin so that it interferes with GLLFILLIN. This interaction can be avoided by enforcing GLLFILLIN after GALSOS. Additionally, GALSOS requires raw pixel values, and ADESPIKE alters the pixel values, so ADESPIKE interferes with this condition. This interaction can be avoided by requiring that GALSOS occur before ADESPIKE.

| | | |
|------------------|-----------|-----------|
| | voyager' | Galileo |
| Execution order: | VGRFILLIN | GALSOS |
| | ADESPIKE | GLLFILLIN |
| | FICOR77 | ADESPIKE |

This simple example illustrates the types of interactions and context-sensitivity that the VICAR image processing application entails. All of these interactions and context sensitive requirements are derived and accounted for automatically by MVP using the operator specification, thus allowing construction of plans despite complex interactions and conditions.

MVP also insulates the user from many of the lower-level intricacies of image processing by automatically achieving subgoals. To illustrate how the operator-based planning process performs subgoaling, consider the subgoal graph illustrated in Figure 7.³ In this case the user has selected the goal that the images be navigated using manual methods and that the archival navigation information for the image should be updated. The decomposition planner has access to the knowledge that in order to navigate the image, the operational goal is to construct an OM matrix which defines the transformation from (line, sample) in the image to some known frame of reference (usually the position relative to the target planet center). The planner knows that in order to compute this matrix it must have a tiepoint file, the project of the image, and the image files formatted into a mosaic file list. In order to produce a tiepoint file for the goal specification of manual navigation, the planner uses the MANMATCH program. The MANMATCH program in turn requires a refined overlap pairs file, the project of the images, the initial predict information, and again a mosaic file list. The refined overlap pairs file can be constructed using the MIBIS program, but this requires a crude overlap pairs file based on an initial predict source. This crude overlap pairs file in turn requires the default navigation method, and the latitude and longitude of sample image files. The rest of the graph is generated similarly. This subgoal graph is generated in response to the particular combination of user goals and the state of the selected image files.

³The VICAR decomposition subgoal graph in Figure 7 is taken from this program's code.

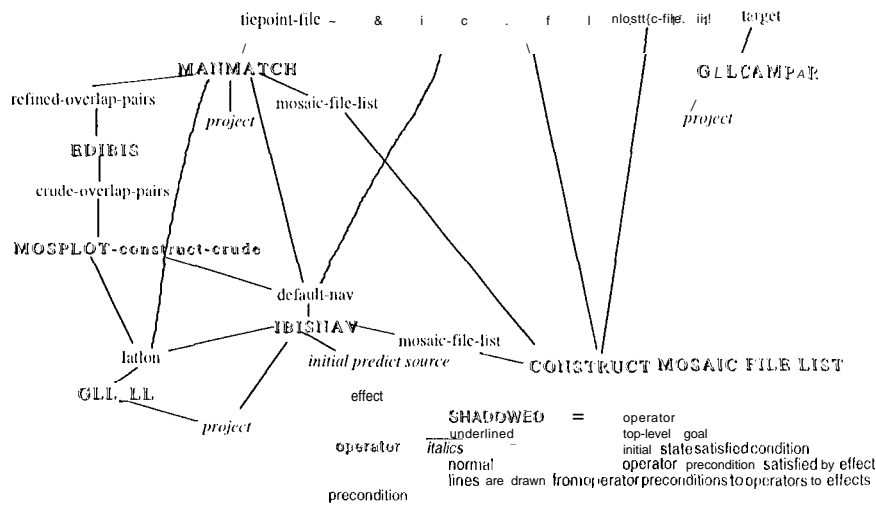


Figure 7: Subgoal Graph for Manual Relative Navigation of Galileo Image Files

MVP also uses operator-based planning techniques to determine correct program option settings. MVP uses preconditions to represent various program option settings and the situations under which they will achieve desired effects. Thus, when an action is selected to achieve a goal, the correct program option settings have also automatically been determined.

3 Status and Impact

MVP2.0 is implemented in C and runs Sun SparcStations under Unix and Motif and 1111(C1 VMS on Vaxes. MVP is currently operational and available for use by analysts at JPL's Multimission Image Processing Laboratory (MIPL) for radiometric correction, color triplet reconstruction, and mosaicking with relative or absolute navigation, registration, and simple filtering and stretching tasks. For these tasks MVP reduces effort to generate an initial PDF for an expert analyst from 1/2 a day to 15 minutes and reduces the effort for a novice analyst from several days to 1 hour. Thus, by using MVP an analyst achieves over an order of magnitude improvement in productivity in generating image processing PDF's.

4 Application Development and Deployment

Initial development on the MVP concept began in August 1992 and a proof of concept demonstration system MVP0 was produced running in 11S1' on a SUN sparystation by late September 1992 for basic correction tasks (radiometric correction, missing line fillin, despiking data, photometric correction, blemish removal and reseau removal) for Voyager project images. This demonstration was invoked using a text-based interface and accessed a dummy image database rather than accessing the actual image database. The complete effort for the proof of concept demonstration was approximately 1 work-

month.

This prototype was well received by the analysts, and work began on a more complete prototype MVP 1.0. This version was also implemented in LISP and accessed a dummy image database in flatfile format. The domain theory for MVP 1.0 extended MVP 0 by including Galileo project images, and added absolute navigation of images as well as several simple filtering and stretching options. MVP 1.0 used a graphical user interface running under Openlook/X. MVP 1.0 was completed by March 1993. MVP 1.0 was further extended to cover application programs for registration of data and simple mosaicking tasks in version 1.1, which was completed in September 1993. The complete development effort for MVP 1.0 and 1.1 was approximately 0.9 work-years, which includes analyst time and MIPS programming support.

In October 1993, work began on MVP 2.0 which was intended to be an operational system in the Multiresolution Image Processing Laboratory (MILIP). From an AI planning standpoint, MVP 2.0 would be very similar to MVP 1.1, with the major difference that it was to be written in C. This included migration of the decomposition planner (to CLIPS) and integration of Flex and Bison to parse input operator and rule files. The majority of the port was completed by March 1994, with April and May being spent on testing, developing a Motif-based GUI, and interfacing MVP 2.0 to actual VICAR database access routines. MVP 2.0 was installed in MILIP, May 1994. During the summer of 1994, MVP 2.0 was extensively tested in the operational setting and used to generate image products. Simultaneously, it was extended to cover more complex correction tasks involving registration, and relative navigation. Also, during this period, a number of knowledge base development tools were produced [Chien 1994b]. The complete development effort for MVP 2.0 from September 1993 through September 1994 was approximately 2.2 work years.

Current efforts focus on two fronts: 1) expanding the domain coverage to further image processing tasks; 2) providing a development environment to facilitate extension to new image processing tasks; and 3) fielding MVP to a University VICAR image processing site (called the Institution Image Processing Sites or IIPS). We are currently working on extending the domain knowledge represented in MVP 2.0 to cover more complex mosaicking tasks as well as filtering and stretching tasks. Development environment enhancements include tools to analyze operator sets and rule sets to find simple errors (e.g. typographical errors) that result in domain theories where no actions can achieve a goal. Towards fielding MVP at a IIPS site, we are currently in contact with personnel from the department of Geology at Arizona State University about a collaborative effort to field MVP for Galileo and Magellan science image processing.

5 Maintenance

Initial development of the planning knowledge base was performed by LISP programmer AI personnel with significant background in AI planning systems (Versions 1.0 and 1.1). The domain theories for Version 2.0 was developed by a software engineer with little AI background. The current domain theory is being used to describe to analysts the overall process of constructing planning decomposition rules and operators, with the intention that further versions of the domain theory will be developed by analysts or other VICAR users. Towards support of this goal, we have been developing a planning knowledge base debugging environment, which provides static analyses of the planning knowledge base to perform simple checks for achievability of goals and runtime tools to

isolate failing preconditions [Chien 1994 b]. In current plans, maintenance and extension of the planner and development environment will be supported by AI group personnel for the near future (e.g., 1-2 years) with the intention that maintenance and extension of MVP1 will eventually be taken over by the Image Processing Section, with the AI group providing continuing support in a consulting role.

6 Conclusions

This paper has described the application of AI planning techniques to automate image processing. This application has resulted in the fielding of MVP2.0, which reduces the effort to produce radiometric correction, color triplet reconstruction, and mosaicking image processing procedures by over an order of magnitude. MVP2.0 uses a hybrid approach to planning, using hierarchical task decomposition and (Jliff' later-1.)sc(1 planning paradigms, as well as traditional syntax translation methods. This successful application is being expanded to cover additional areas of image processing and fielding to remote university image processing sites.

References

- [Chapman 1987] D. Chapman, "Planning for Conjunctive Goals, 1987, " *Artificial Intelligence* 32, 3.
- [Chien 1994a] S. Chien, "Using AI Planning Techniques to Automatically Generate Image Processing Procedures: A Preliminary Report," Proceedings of the Second International Conference on AI Planning Systems, Chicago, IL, June 1994, pp. 215-224.
- [Chien 1994b] S. Chien, "Towards an Intelligent Planning Knowledge-base Development Environment," Proceedings of the 1994 AAAI Fall Symposium on Learning and Planning: On to Real Applications, New Orleans, LA, November 1994, pp. 23-27.
- [Chien 1994c] S. Chien, "Automated Synthesis of Image Processing Procedures for a Large-scale Image Database," Proceedings of the First IEEE International Conference on Image Processing, Austin, TX, November 1994, Vol. 3, pp. 796-800.
- [Erol et al. 1994] K. Erol, J. Hendler, and D. Nau, "UMCP: A Sound and Complete Procedure for Hierarchical Task Network Planning," Proceedings of the Second International Conference on AI Planning Systems, Chicago, IL, June 1994, pp. 249-254.
- [Iwasaki and Friedland 1985] Y. Iwasaki and P. Friedland, "The Concept and Implementation of Skeletal Plans," *Journal of Automated Reasoning* 1, 1 (1985), pp. 161-208.
- [Lansky 1993] A. Lansky, Localized Planning with Diverse Plan Construction Methods, Technical Report F1A-93-17, NASA Ames Research Center, June 1993.
- [LaVoie et al. 1989] S. LaVoie, D. Alexander, C. Avis, H. Mortensen, C. Stanley, and L. Wainio, VICAR User's Guide, Version 2, JPL Internal Document D-4186, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 1989.
- [Pemberthy and Weld 1992] J. S. Pemberthy and D. S. Weld, "UCPOP: A Sound Complete, Partial Order Planner for A DL," Proceedings of the Third International Conference on Knowledge Representation and Reasoning, October 1992.
- [Stefik 1981] M. Stefik, "Planning with Constraints (MOLGEN: Part I)," *Artificial Intelligence* 16,2(1981), pp. 111-140.